

# Disciplina BCM0505-15

## Processamento da Informação

Introdução ao Python

---

**Profa. Carla Negri Lintzmayer**

carla.negri@ufabc.edu.br

<http://professor.ufabc.edu.br/~carla.negri>

Centro de Matemática, Computação e Cognição  
Universidade Federal do ABC



# Conteúdo deste conjunto de slides

Python

Preparando o ambiente

Variáveis

Entrada e saída em Python

Expressões aritméticas em Python

Expressões relacionais em Python

Expressões lógicas em Python

Comentários em Python

Pratique!

# Python

---



# Por que Python?

- Facilita o aprendizado.
- Legibilidade do código.
- Software livre.
- Utilizada em várias áreas (mercado de trabalho e comunidade científica).

# Fluência em Python?

- Lembre-se que essa disciplina não tem por objetivo *ensinar Python*.
  - Mesmo se fosse, não teríamos tempo.
  - Python é uma linguagem muito completa, cheia de bibliotecas que nos ajudam a acessar bancos de dados, processar arquivos das mais variadas extensões, construir interfaces gráficas, e muitas outras!
- Nós vamos usar Python como ferramenta para implementar os algoritmos que vamos criar.
  - Criar algoritmos usando lógica de programação é nosso objetivo!

# Preparando o ambiente

---

# Instalação

- Para usar Python, precisamos primeiro de um **interpretador** da linguagem.
- Ele aceita comandos escritos em Python e os executa, linha a linha.
- É ele quem traduz os programas para algo que possa ser executado pelo computador.
- Poderíamos instalar o Python diretamente do site oficial, porém existe uma distribuição chamada Anaconda que já instala o Python e alguns outros programas úteis: [faça o download aqui](#).
- Certifique-se de estar baixando o arquivo para o Sistema Operacional correto e que a versão do Python é **3.X**, onde X pode ser 7 ou 8.

# Spyder

Após a instalação, vamos abrir um programa que vem junto com o Anaconda chamado **Spyder**.



# Spyder

The image shows the Spyder Python IDE interface. On the left, a script editor displays a temporary file with the following content:

```
1 # -*- coding: utf-8 -*-  
2 """"  
3 Spyder Editor  
4  
5 This is a temporary script file.  
6 """"  
7  
8
```

A large white number '1' is overlaid on the left side of the script editor.

On the right, a help window titled 'Usage' is open, displaying the following text:

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

A large white number '2' is overlaid on the help window.

Below the help window, the console window is visible, showing the following output:

```
Python 3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916  
64 bit (AMD64)]  
Type "copyright", "credits" or "license" for more  
information.  
  
IPython 7.22.0 -- An enhanced Interactive Python.  
  
In [1]:
```

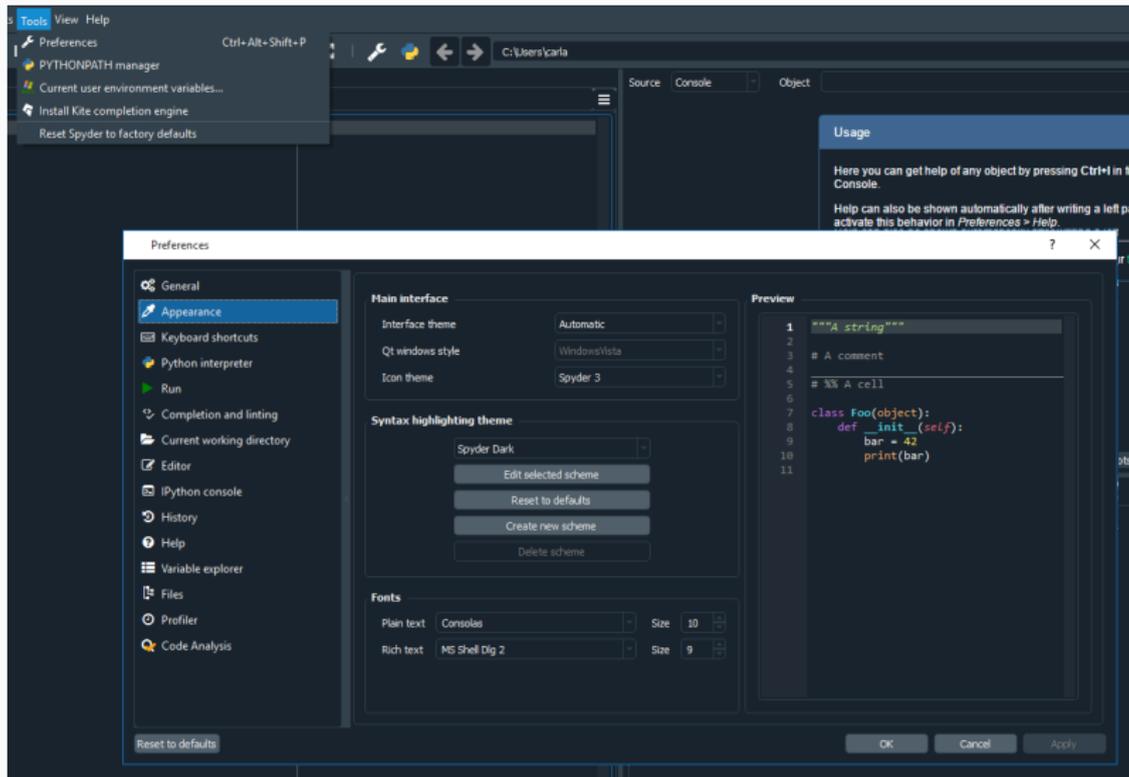
A large white number '3' is overlaid on the console window.

The bottom status bar of the IDE shows: 'Python console | History', 'LSP Python | ready', 'conda Python 3.8.8', 'Line 1, Col 1', 'UTF-8', 'CRLF', 'File', and 'View 40%'.

Na figura anterior:

1. É o editor de texto, onde escreveremos as instruções dos nossos programas;
2. É uma janela de auxílio, onde você conseguirá ter acesso à diferentes informações que podem facilitar o desenvolvimento do seu programa;
3. É o *console* Python, onde você poderá inserir diretamente algumas instruções para ser executadas instantaneamente e poderá ver a saída dos seus programas.

Configure a aparência como você preferir.



## Outra opção

Ferramenta online:

[replit.com](https://replit.com)

# Variáveis

---

# Variáveis em Python

- Identificadores de variáveis: só podem conter letras, números e o símbolo “\_”, nunca começando com número.
- Em Python, não há necessidade de declarar variáveis explicitamente: isso é feito implicitamente na hora de uma atribuição.
- Por isso, a tipagem em Python é dita *dinâmica*.
  - É possível saber o valor de uma variável usando o comando `type(identificador)`.
  - Python tem os tipos básicos, `int` (inteiro), `float` (real), `str` (string), `bool` (lógico), e muitos outros (veremos alguns mais adiante no curso).
- Sintaxe da atribuição:

```
identificador = expressao
```

# Entrada e saída em Python

---

# Leitura e escrita

- Usaremos as *funções* `input()` para leitura e `print()` para escrita.
- A função `input()` lê todo conteúdo digitado pelo usuário até encontrar uma quebra de linha (o usuário digitou “Enter”) e devolve uma *string* com esse conteúdo.
- Usaremos, em conjunto com `input()` a função `split()`, que “quebra” uma string de acordo com os espaços contidos nela.
- A sintaxe básica é:

```
identificador = input()
```

# Leitura com input()

---

```
1  >>> entrada = input()
2  3 65 8.3 texto 4
3  >>> entrada
4  '3 65 8.3 texto 4'
5  >>> entrada.split()
6  ['3', '65', '8.3', 'texto', '4']
7  >>> entrada = entrada.split()
8  >>> entrada
9  ['3', '65', '8.3', 'texto', '4']
10 >>> entrada[1]
11 '65'
12 >>> int(entrada[1])
13 65
14 >>> entrada[2]
15 '8.3'
16 >>> float(entrada[2])
17 8.3
18 >>> entrada[3]
19 'texto'
```

# Lendo vários números de uma única linha

Se forem números inteiros:

---

```
1 linha = input().split()
2
3 num1 = int(linha[0])
4 num2 = int(linha[1])
5 ...
```

---

Se forem números reais:

---

```
1 linha = input().split()
2
3 num1 = float(linha[0])
4 num2 = float(linha[1])
5 ...
```

---

# Lendo um número por linha

Se forem inteiros:

---

```
1 num1 = int(input())
2 num2 = int(input())
3 ...
```

---

Se forem reais:

---

```
1 num1 = float(input())
2 num2 = float(input())
3 ...
```

---

- A função `print()` escreve algo na tela.
- A sintaxe básica é:

```
print(expressao)
```

## Escrevendo dados com `print()`

---

```
1 print(5)
2 print("texto com nova linha")
3 print("texto sem nova linha", end='')
4 a = 5
5 b = 156
6 print("imprimindo inteiro: a = %d e b = %d" % (a, b))
7 a = 5.55555
8 b = 3.14156
9 print("imprimindo real: a = %f e b = %.3f" % (a, b))
10 a = "carla"
11 print("O nome da professora eh " + a)
12 print("O nome da professora eh", a)
13 print("O nome da professora eh %s" % (a))
```

---

# Expressões aritméticas em Python

---

# Expressões aritméticas

Operadores aritméticos, por ordem de prioridade:

- parênteses
- \*\* (exponenciação)
- - (inverso aditivo)
- \* (multiplicação), / (divisão), // (divisão inteira), % (resto da divisão inteira)
- + (soma), - (subtração)

Na dúvida, use parênteses.

# Strings

- Strings em Python são qualquer conjunto de caracteres que estejam dentro de aspas.
- O operador + funciona como um concatenador de strings: transforma em uma única string as duas que estiverem ao seu redor.
- O operador \* funciona como um concatenador de uma string com ela mesma, um certo número de vezes.

---

```
1 nome = "Carla"
2 sobrenome1 = "Negri"
3 sobrenome2 = "Lintzmayer"
4 nome_completo = nome + sobrenome1 + sobrenome2
5 print(nome_completo)
6 nome_completo = nome + " " + sobrenome1 + " " + sobrenome2
7 print(nome_completo)
```

# Expressões relacionais em Python

---

Os operadores relacionais são:

- $==$  (igual a)
- $>$  (maior que)
- $<$  (menor que)
- $\geq$  (maior ou igual a)
- $\leq$  (menor ou igual a)
- $\neq$  (diferente de)

# Expressões relacionais

**ATENÇÃO!** Não confundir o operador de comparação de igualdade (`==`) com o de atribuição (`=`).

---

```
1 print(4-1 <= 2)
2 print(3.5/3 == 1)
3 print(3.5/3 = 1)
4 x = 3
5 print(2**3 != 4*2)
```

---

# Expressões lógicas em Python

---

# Expressões lógicas

Os operadores lógicos são:

- and: conjunção
- or: disjunção
- not: negação

---

```
1 frio = input()
2 chuva = input()
3 tempoRuim = frio == "s" or chuva == "s"
4 print(tempoRuim)
```

---

# Comentários em Python

---

# Comentários

- Qualquer conteúdo entre um símbolo `#` e o final da linha será ignorado pelo interpretador.
- Qualquer conteúdo entre `"""` e `"""` também.
- Usamos isso para colocar comentários explicativos do código.
- Eles servem para ajudar ao próprio criador do código, quando volta a vê-lo algum tempo depois de tê-lo feito.
- Também servem para outras pessoas que precisem ler o código.
- Comentários não devem ser grandes demais, pois o código deve ser simples o suficiente para que blocos dele possam ser compreendidos.
  - Colocamos então comentários nesses blocos.

**Pratique!**

---

- Moodle
- Exercícios de apoio
- URI